JavaScript and Cookies

What are Cookies ?

Web Browsers and Servers use HTTP protocol to communicate and HTTP is a stateless protocol. But for a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages. But how to maintain users' session information across all the web pages.

In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

How It Works ?

Your server sends some data to the visitor's browser in the form of a cookie. The browser may accept the cookie. If it does, it is stored as a plain text record on the visitor's hard drive. Now, when the visitor arrives at another page on your site, the browser sends the same cookie to the server for retrieval. Once retrieved, your server knows/remembers what was stored earlier.

Cookies are a plain text data record of 5 variable-length fields −

- **Expires** − The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- **Domain** − The domain name of your site.
- **Path** − The path to the directory or web page that set the cookie. This may be blank if you want to retrieve the cookie from any directory or page.
- **Secure** − If this field contains the word "secure", then the cookie may only be retrieved with a secure server. If this field is blank, no such restriction exists.
- **Name=Value** − Cookies are set and retrieved in the form of key-value pairs

Cookies were originally designed for CGI programming. The data contained in a cookie is automatically transmitted between the web browser and the web server, so CGI scripts on the server can read and write cookie values that are stored on the client.

JavaScript can also manipulate cookies using the **cookie** property of the **Document** object. JavaScript can read, create, modify, and delete the cookies that apply to the current web page.

Storing Cookies

The simplest way to create a cookie is to assign a string value to the document.cookie object, which looks like this.

```
document.cookie = "key1 = value1;key2 = value2;expires = date";
```

Here the **expires** attribute is optional. If you provide this attribute with a valid date or time, then the cookie will expire on a given date or time and thereafter, the cookies' value will not be accessible.

Note − Cookie values may not include semicolons, commas, or whitespace. For this reason, you may want to use the JavaScript **escape()** function to encode the value before storing it in

the cookie. If you do this, you will also have to use the corresponding **unescape()** function when you read the cookie value.

Example

Try the following. It sets a customer name in an input cookie.

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function WriteCookie() {
          if( document.myform.customer.value == "" ) {
            alert("Enter some value!");
            return;
          }
          cookievalue = escape(document.myform.customer.value) + ";";
          document.cookie = "name=" + cookievalue;
          document.write ("Setting Cookies : " + "name=" + cookievalue );
        }
      //-->
    </script>
  </head>

  <body>
    <form name = "myform" action = "">
      Enter name: <input type = "text" name = "customer"/>
      <input type = "button" value = "Set Cookie" onclick = "WriteCookie();"/>
    </form>
  </body>
</html>
```

Output

Now your machine has a cookie called **name**. You can set multiple cookies using multiple key = value pairs separated by comma.

Reading Cookies

Reading a cookie is just as simple as writing one, because the value of the document.cookie object is the cookie. So you can use this string whenever you want to access the cookie. The document.cookie string will keep a list of name=value pairs separated by semicolons, where **name** is the name of a cookie and value is its string value.

You can use strings' **split()** function to break a string into key and values as follows −

Example

Try the following example to get all the cookies.

```
<html>
```

```
<head>
  <script type = "text/javascript">
    <!--
      function ReadCookie() {
        var allcookies = document.cookie;
        document.write ("All Cookies : " + allcookies );

        // Get all the cookies pairs in an array
        cookiearray = allcookies.split(';');

        // Now take key value pair out of this array
        for(var i=0; i<cookiearray.length; i++) {
          name = cookiearray[i].split('=')[0];
          value = cookiearray[i].split('=')[1];
          document.write ("Key is : " + name + " and Value is : " + value);
        }
      }
    //-->
  </script>
</head>

<body>
  <form name = "myform" action = "">
    <p> click the following button and see the result:</p>
    <input type = "button" value = "Get Cookie" onclick = "ReadCookie()"/>
  </form>
</body>
</html>
```

Note − Here **length** is a method of **Array** class which returns the length of an array. We will discuss Arrays in a separate chapter. By that time, please try to digest it.

Note − There may be some other cookies already set on your machine. The above code will display all the cookies set on your machine.

Setting Cookies Expiry Date

You can extend the life of a cookie beyond the current browser session by setting an expiration date and saving the expiry date within the cookie. This can be done by setting the **'expires'** attribute to a date and time.

Example

Try the following example. It illustrates how to extend the expiry date of a cookie by 1 Month.

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function WriteCookie() {
          var now = new Date();
          now.setMonth( now.getMonth() + 1 );
```

```
            cookievalue = escape(document.myform.customer.value) + ";"

            document.cookie = "name=" + cookievalue;
            document.cookie = "expires=" + now.toUTCString() + ";"
            document.write ("Setting Cookies : " + "name=" + cookievalue );
         }
        //-->
      </script>
   </head>

   <body>
      <form name = "myform" action = "">
         Enter name: <input type = "text" name = "customer"/>
         <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
      </form>
   </body>
</html>
```

Output

Deleting a Cookie

Sometimes you will want to delete a cookie so that subsequent attempts to read the cookie return nothing. To do this, you just need to set the expiry date to a time in the past.

Example

Try the following example. It illustrates how to delete a cookie by setting its expiry date to one month behind the current date.

```
<html>
   <head>
      <script type = "text/javascript">
         <!--
            function WriteCookie() {
               var now = new Date();
               now.setMonth( now.getMonth() - 1 );
               cookievalue = escape(document.myform.customer.value) + ";"

               document.cookie = "name=" + cookievalue;
               document.cookie = "expires=" + now.toUTCString() + ";"
               document.write("Setting Cookies : " + "name=" + cookievalue );
            }
         //-->
      </script>
   </head>

   <body>
      <form name = "myform" action = "">
         Enter name: <input type = "text" name = "customer"/>
```

```
        <input type = "button" value = "Set Cookie" onclick = "WriteCookie()"/>
      </form>
   </body>
</html>
```

Output